RÉFÉRENCE RAPIDE KUBERNETES

kubectl, pods, déploiements, services, configs, débogage

Bases kubectl

Infos cluster

```
kubectl cluster-info
kubectl get nodes
kubectl config current-context
kubectl config use-context my-cluster
```

Commandes essentielles

`kubectl get <resource>`	Lister les ressources
`kubectl describe <resource> <name>`	Informations détaillées sur la ressource
`kubectl create -f file.yaml`	Créer une ressource depuis un fichier
`kubectl apply -f file.yaml`	Créer ou mettre à jour une ressource
`kubectl delete -f file.yaml`	Supprimer une ressource depuis un fichier
`kubectl edit <resource> <name>`	Modifier une ressource en place
`kubectl api-resources`	Lister tous les types de ressources

Formats de sortie

`-o wide`	Colonnes supplémentaires (IP, nœud)
`-o yaml`	Sortie YAML complète
`-o json`	Sortie JSON complète
`-o jsonpath='{.spec}'`	Extraire des champs spécifiques
`--sort-by=.metadata.name`	Trier la sortie par champ

Pods

Opérations sur les pods

```
kubectl get pods
kubectl get pods -A                 # all namespaces
kubectl run nginx --image=nginx     # quick pod
kubectl delete pod nginx
```

YAML de pod

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp
  labels: { app: myapp }
spec:
  containers:
    - name: app
      image: nginx:1.27
      ports:
        - containerPort: 80
```

Valeurs d'état des pods

`Running`	Tous les conteneurs démarrés
`Pending`	En attente de planification ou de téléchargement d'image
`CrashLoopBackOff`	Le conteneur plante et redémarre en boucle
`ImagePullBackOff`	Impossible de télécharger l'image du conteneur
`Completed`	Exécution terminée (Jobs)

Déploiements

YAML de déploiement

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels: { app: web }
  template:
    metadata:
      labels: { app: web }
    spec:
      containers:
        - name: web
          image: nginx:1.27
          ports:
            - containerPort: 80
```

Commandes de déploiement

`kubectl get deploy`	Lister les déploiements
`kubectl scale deploy web --replicas=5`	Mettre à l'échelle les répliques
`kubectl set image deploy/web web=nginx:1.28`	Mettre à jour l'image (déploiement progressif)
`kubectl rollout status deploy/web`	Surveiller la progression du déploiement
`kubectl rollout undo deploy/web`	Revenir à la révision précédente
`kubectl rollout history deploy/web`	Voir l'historique des révisions

Services

Types de service

`ClusterIP`	Interne uniquement (défaut)
`NodePort`	Exposer sur l'IP de chaque nœud à un port statique
`LoadBalancer`	Équilibreur de charge externe (cloud)
`ExternalName`	Alias DNS vers un service externe

YAML de service

```
apiVersion: v1
kind: Service
metadata:
  name: web-svc
spec:
  type: ClusterIP
  selector: { app: web }
  ports:
    - port: 80
      targetPort: 80
```

Exposition rapide

```
kubectl expose deploy web --port=80 --type=ClusterIP
kubectl expose deploy web --port=80 --type=NodePort
kubectl get svc
```

ConfigMaps & Secrets

ConfigMap

```
kubectl create configmap app-cfg \
  --from-literal=DB_HOST=db.example.com \
  --from-file=config.ini
```

Secret

```
kubectl create secret generic db-creds \
  --from-literal=username=admin \
  --from-literal=password=s3cret
```

Utiliser dans les pods

```
# As environment variables
envFrom:
  - configMapRef: { name: app-cfg }
  - secretRef:    { name: db-creds }

# As volume mount
volumes:
  - name: cfg
    configMap: { name: app-cfg }
```

Commandes

`kubectl get cm`	Lister les ConfigMaps
`kubectl get secret`	Lister les Secrets
`kubectl describe cm app-cfg`	Afficher les données du ConfigMap
`kubectl get secret db-creds -o yaml`	Afficher le Secret (encodé en base64)

Namespaces

Commandes de namespace

`kubectl get ns`	Lister les namespaces
`kubectl create ns staging`	Créer un namespace
`kubectl delete ns staging`	Supprimer le namespace et toutes ses ressources
`kubectl get pods -n staging`	Lister les pods dans un namespace
`kubectl get pods -A`	Lister les pods dans tous les namespaces

Définir le namespace par défaut

```
kubectl config set-context --current \
  --namespace=staging
```

Volumes

PersistentVolumeClaim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: data-pvc
spec:
  accessModes: [ReadWriteOnce]
  resources:
    requests: { storage: 10Gi }
```

Monter dans un pod

```
volumes:
  - name: data
    persistentVolumeClaim:
      claimName: data-pvc
containers:
  - volumeMounts:
      - name: data
        mountPath: /app/data
```

Types de volume

`emptyDir`	Répertoire temporaire, supprimé avec le pod
`hostPath`	Monter un chemin du système hôte
`persistentVolumeClaim`	Stockage persistant (PVC)
`configMap`	Monter un ConfigMap en tant que fichiers
`secret`	Monter un Secret en tant que fichiers

Ingress

YAML Ingress

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: web-ingress
spec:
  rules:
    - host: app.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web-svc
                port: { number: 80 }
```

Notes Ingress

`Ingress Controller`	Obligatoire (nginx-ingress, traefik, etc.)
`pathType: Prefix`	Correspondance par préfixe d'URL
`pathType: Exact`	Correspondance exacte du chemin URL
`TLS`	Ajouter une section `tls:` avec le nom du secret

Débogage

Commandes de diagnostic

`kubectl logs <pod>`	stdout/stderr du conteneur
`kubectl logs <pod> -c <ctr>`	Logs d'un conteneur spécifique
`kubectl logs <pod> --previous`	Logs du conteneur planté
`kubectl describe pod <pod>`	Événements, conditions, statut
`kubectl exec -it <pod> -- sh`	Shell dans le conteneur
`kubectl port-forward <pod> 8080:80`	Rediriger un port local vers le pod
`kubectl top pods`	Utilisation CPU/mémoire (metrics-server)
`kubectl get events --sort-by=.lastTimestamp`	Chronologie des événements du cluster

Pod de débogage

```
kubectl run debug --rm -it --image=busybox -- sh
# or attach ephemeral container
kubectl debug -it <pod> --image=busybox
```

Patterns courants

Labels & Sélecteurs

```
kubectl get pods -l app=web
kubectl get pods -l 'env in (prod,staging)'
kubectl label pod myapp env=prod
```

Limites de ressources

```
resources:
  requests: { cpu: 100m, memory: 128Mi }
  limits:   { cpu: 500m, memory: 256Mi }
```

Liveness & Readiness

```
livenessProbe:
  httpGet: { path: /healthz, port: 8080 }
  initialDelaySeconds: 5
  periodSeconds: 10
readinessProbe:
  httpGet: { path: /ready, port: 8080 }
```

Recettes rapides

`Simulation à sec`	`kubectl apply -f file.yaml --dry-run=client`
`Générer du YAML`	`kubectl create deploy web --image=nginx --dry-run=client -o yaml`
`Surveiller`	`kubectl get pods -w`
`Copier des fichiers`	`kubectl cp file.txt pod:/tmp/`
`Redémarrer un déploiement`	`kubectl rollout restart deploy/web`